```
/***********************************************************************
* Program:
*    Project 07, Calendar
*    Brother Ridges, CS124
* Author:
*    Erik Rybalkin
* Summary:
*    This program prompts a user for year and month values, and
*    converts that data into a well formatted table of dates.
*
*    Estimated:  3.0 hrs
*    Actual:     4.20 hrs
*      Didn't have problems with this program
************************************************************************/
#include <iostream>
#include <iomanip>
using namespace std;

int getMonth();
int getYear();
bool isLeapYear(int year);
int monthCount(int month, int year);
int yearCount(int year);
int getOffset(int month, int year);
int getDaysInMonth(int month, int year);
void displayYear(int year);
void displayMonth(int month);
void displayTable(int offset, int numDays);



/***********************************************************************
 * This function is an entry point of the program
 ***********************************************************************/
int main[RS1] ()
{
   int month = getMonth();
   int year = getYear();

   int daysInMonth = getDaysInMonth(month, year);

   displayMonth(month);
   displayYear(year);
   displayTable(getOffset(month, year), daysInMonth);

   return 0;

}


/***********************************************************************
 * This function prompts a user for a month
 ***********************************************************************/
int getMonth()
{
   int month;

   while (month < 1 || month > 12)
   {
      cout << "Enter a month number: ";
      cin  >> month;

      if (month < 1 || month > 12)
         cout << "Month must be between 1 and 12.\n";
   }

   return month;
}

/***********************************************************************
 * This function prompts a user for a year.
 ***********************************************************************/
int getYear()
{
   int year;

   while (year < 1753)
   {
      cout << "Enter year: ";
      cin  >> year;
      if (year < 1753)
         cout << "Year must be 1753 or later.\n";
   }
   cout << endl;
   return year;
}




/**********************************************************************
 * Get the amount of days in a particular month.
 ***********************************************************************/
int monthCount(int month, int year)
{
   int daysInMonth = 0;

   for (int initialMonth = 1; initialMonth < month; initialMonth++)
   {
      if (initialMonth == 1 || initialMonth == 3 || initialMonth == 5
          || initialMonth == 7 || initialMonth == 8
          || initialMonth == 10 || initialMonth == 12)
         daysInMonth += 31;

      else if (initialMonth == 4 || initialMonth == 6 ||
              initialMonth == 9 || initialMonth == 11)
         daysInMonth += 30;

      else if (initialMonth == 2 && !isLeapYear(year))
         daysInMonth += 28;

      else
         daysInMonth += 29;

   }

   return daysInMonth;
}

/**********************************************************************
 * This function gets the total number of days from 1753 to a selected year.
 ***********************************************************************/
int yearCount(int year)
{
   int yearDays = 0;

   for (int previousYear = 1753; previousYear < year; previousYear++)
   {
      if (!isLeapYear(previousYear))
         yearDays += 365;

      if (isLeapYear(previousYear))
         yearDays += 366;
   }

   return yearDays;
}

/**********************************************************************
 * This function gets the offset - spacing to visualize on which day of
 * the week month starts.
 ***********************************************************************/
int getOffset(int month, int year)
{
   int yearDays = yearCount(year);

   int monthDays = monthCount(month, year);

   int offset = (yearDays + monthDays) % 7;

   return offset;
}


/**********************************************************************
 * This function gets the exact number of days for a selected month.
 ***********************************************************************/
int getDaysInMonth(int month, int year)
{
   int daysInMonth;

   if (month == 1 || month == 3 || month == 5
         || month == 7 || month == 8
         || month == 10 || month == 12)
      daysInMonth = 31;
   else if (month == 4 || month == 6 ||
            month == 9 || month == 11)
      daysInMonth = 30;

   else if (month == 2 && !isLeapYear(year))
      daysInMonth = 28;

   else
      daysInMonth = 29;

   return daysInMonth;
}


/**********************************************************************
 * This function displays a calendar data.
 ***********************************************************************/
void displayTable(int offsetting, int daysInMonth)
{

   cout << "  Su  Mo  Tu  We  Th  Fr  Sa\n";

   int day;

   if (offsetting == 0)
   {
      day = 2;
      cout << setw(6);
   }
   else if (offsetting == 1)
   {
      day = 3;
      cout << setw(10);
   }
   else if (offsetting == 2)
   {
      day = 4;
      cout << setw(14);
   }
   else if (offsetting == 3)
   {
      day = 5;
      cout << setw(18);
   }
   else if (offsetting == 4)
   {
      day = 6;
      cout << setw(22);
   }
   else if (offsetting == 5)
   {
      day = 7;
      cout << setw(26);
   }
   else if (offsetting == 6)
   {
      day = 1;
      cout << setw(2);
   }
   else
      ;

   for (int weeksDay = 1; weeksDay <= daysInMonth; weeksDay++)
   {
      cout << "  " << setw(2) << weeksDay;
      day++;
      if (day == 8)
      {
         cout << endl;
         day = 1;
      }
   }

   if (day >= 2 && day <= 8)
      cout << endl;

   return;

}


/**********************************************************************
 * This function displays a year.
 ***********************************************************************/
void displayYear(int year)
{
   cout << ", " << year << endl;
```

```
}

/***********************************************************************
 * Convert a month's number into a month string and display it.
 ***********************************************************************/
void displayMonth(int month)
{
   switch (month)
   {
      case 1 :
         cout << "January";
         break;
      case 2 :
         cout << "February";
         break;
      case 3 :
         cout << "March";
         break;
      case 4 :
         cout << "April";
         break;
      case 5 :
         cout << "May";
         break;
      case 6 :
         cout << "June";
         break;
      case 7 :
         cout << "July";
         break;
      case 8 :
         cout << "August";
         break;
      case 9 :
         cout << "September";
         break;
      case 10 :
         cout << "October";
         break;
      case 11 :
         cout << "November";
         break;
      case 12 :
         cout << "December";
         break;
   }
}

/***********************************************************************
 * This function checks if a current year is a leap year.
 ***********************************************************************/
bool isLeapYear(int year)
{
   return bool (year % 4 == 0 && year % 100 != 0 || year % 400 == 0);
}
```

Style Checker Results

```
CORRECT SPELLING:
   Su
   Erik
   Tu
   Rybalkin
   Sa
   Th

CHECK CORRECT CASE USING THIS LIST:

NAME          Line #
----          ------

VARIABLES:
getMonth       19
getYear        20
year           21, 22, 23, 24, 25, 26, 38, 75, 95, 124, 144, 159, 252, 306
month          22, 24, 25, 27, 37, 56, 95, 144, 159, 260
numDays        28
offset         28, 150
daysInMonth    40, 97, 161, 185
initialMonth   99
yearDays       126, 146
previousYear   128
monthDays      148
offsetting     185
```

```
day            190
weeksDay        230

SUBROUTINES:
main           35
getMonth        54
getYear        73
monthCount      95
yearCount       124
getOffset       144
getDaysInMonth  159
displayTable    185
displayYear     252
displayMonth    260
isLeapYear      306
```

Test Bed Results

a.out:

```
------------------------------------------------------------
Starting Test 1

Trivial case; January 1st, 1753 is a Monday. Here the offset is 0

   > Enter a month number: 1
   > Enter year: 1753
   >
   > January, 1753
   >   Su  Mo  Tu  We  Th  Fr  Sa
   >        1   2   3   4   5   6
   >    7   8   9  10  11  12  13
   >   14  15  16  17  18  19  20
   >   21  22  23  24  25  26  27
   >   28  29  30  31

Test 1 passed.
------------------------------------------------------------

------------------------------------------------------------
Starting Test 2

February 1st, 1753 is 31 days away from January 1st.

   > Enter a month number: 2
   > Enter year: 1753
   >
   > February, 1753
   >   Su  Mo  Tu  We  Th  Fr  Sa
   >                    1   2   3
   >    4   5   6   7   8   9  10
   >   11  12  13  14  15  16  17
   >   18  19  20  21  22  23  24
   >   25  26  27  28

Test 2 passed.
------------------------------------------------------------

------------------------------------------------------------
Starting Test 3

February 1753 is not a leap year so there are 28 days,

   > Enter a month number: 3
   > Enter year: 1753
   >
   > March, 1753
   >   Su  Mo  Tu  We  Th  Fr  Sa
   >                    1   2   3
   >    4   5   6   7   8   9  10
   >   11  12  13  14  15  16  17
   >   18  19  20  21  22  23  24
   >   25  26  27  28  29  30  31

Test 3 passed.
------------------------------------------------------------

------------------------------------------------------------
Starting Test 4

This test is to make sure that the correct number of days for each month
is in the program.

   > Enter a month number: 12
   > Enter year: 1753
```

```
>
> December, 1753
>   Su  Mo  Tu  We  Th  Fr  Sa
>                            1
>    2   3   4   5   6   7   8
>    9  10  11  12  13  14  15
>   16  17  18  19  20  21  22
>   23  24  25  26  27  28  29
>   30  31
```

Test 4 passed.

Starting Test 5

This test is to check add years when leap years are not involved

```
> Enter a month number: 1
> Enter year: 1755
>
> January, 1755
>   Su  Mo  Tu  We  Th  Fr  Sa
>                1   2   3   4
>    5   6   7   8   9  10  11
>   12  13  14  15  16  17  18
>   19  20  21  22  23  24  25
>   26  27  28  29  30  31
```

Test 5 passed.

Starting Test 6

Even though 1756 is a leap year, we should not count 356 days when
adding the days for the month. It only counts after the 28th of February, 1756

```
> Enter a month number: 1
> Enter year: 1756
>
> January, 1756
>   Su  Mo  Tu  We  Th  Fr  Sa
>                    1   2   3
>    4   5   6   7   8   9  10
>   11  12  13  14  15  16  17
>   18  19  20  21  22  23  24
>   25  26  27  28  29  30  31
```

Test 6 passed.

Starting Test 7

Leap year

```
> Enter a month number: 2
> Enter year: 1756
>
> February, 1756
>   Su  Mo  Tu  We  Th  Fr  Sa
>    1   2   3   4   5   6   7
>    8   9  10  11  12  13  14
>   15  16  17  18  19  20  21
>   22  23  24  25  26  27  28
>   29
```

Test 7 passed.

Starting Test 8

Not a leap year

```
> Enter a month number: 2
> Enter year: 1800
>
> February, 1800
>   Su  Mo  Tu  We  Th  Fr  Sa
>                            1
>    2   3   4   5   6   7   8
>    9  10  11  12  13  14  15
>   16  17  18  19  20  21  22
>   23  24  25  26  27  28
```

```
Test 8 passed.
------------------------------------------------------------

------------------------------------------------------------
Starting Test 9

Leap year

  > Enter a month number: 2
  > Enter year: 2000
  >
  > February, 2000
  >   Su  Mo  Tu  We  Th  Fr  Sa
  >            1   2   3   4   5
  >    6   7   8   9  10  11  12
  >   13  14  15  16  17  18  19
  >   20  21  22  23  24  25  26
  >   27  28  29

Test 9 passed.
------------------------------------------------------------

------------------------------------------------------------
Starting Test 10

Your program should be able to handle invalid input for days and for years.
In this case, it should simply re-prompt the user for valid data.

  > Enter a month number: 13
  > Month must be between 1 and 12.
  > Enter a month number: -1
  > Month must be between 1 and 12.
  > Enter a month number: 11
  > Enter year: 90
  > Year must be 1753 or later.
  > Enter year: -1
  > Year must be 1753 or later.
  > Enter year: 2002
  >
  > November, 2002
  >   Su  Mo  Tu  We  Th  Fr  Sa
  >                        1   2
  >    3   4   5   6   7   8   9
  >   10  11  12  13  14  15  16
  >   17  18  19  20  21  22  23
  >   24  25  26  27  28  29  30

Test 10 passed.
------------------------------------------------------------


============================================================
Passed [RS2] all tests with no errors.
============================================================
```

Grading Criteria

Criteria	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%	Weight	Score
computeOffset()	The code is elegant and efficient	Correctly computes the offset for all input	One bug exists	Elements of the solution exist	No attempt was made to implement the computeOffset() function	30	100
display()	No test bed errors	Looks good on screen	One serious formatting error	Some calendar data is displayed	Function missing	30	100
isLeapYear()	The code is elegant and efficient	Correctly computes the leap year for all input	One bug exists	Elements of the solution exist	No attempt was made to implement the isLeapYear() function	10	100
Modularization	Function design is well thought-out and elegant	All functions work correctly	One bug calling a function or unnecessary data passed between functions	Multiple functions exist in the project	Only one function is used	20	100
Style	Great variable names, no errors, great	No style checker errors	A few style checker errors	Misleading variable names or gross style	Little effort was spent on style	10	100

	comments			errors			
Total							100

ryb16001@byui.edu

[RS1] Good layout. Your code is easy to read.
[RS2] Good job!